Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 30 (2012) 678 – 685

Procedia Engineering

www.elsevier.com/locate/procedia

International Conference on Communication Technology and System Design 2011

An Innovative Routing Technique to Optimize Time and Speed

Brindha G.R[a] , Anand.S[c], Gosakan.V[c] and Joe Prathap.P.M [b] ,a*

[c]*Mechatronics Dept, SASTRA University, Thanjavur, Tamil Nadu,India*

[a]*ICT Dept, School of Computing, SASTRA University, Thanjavur, Tamil Nadu,India*

[b]*IT Dept, RMD Engineering College,R.S.M. Nagar, Kavaraipettai, Tiruvallur District, India.*

Abstract

In this paper we introduce a novel algorithm, MAXSMINT (Maximizing Speed and Minimizing Time) for determining the shortest route in a railway network. In today's busy world, no one takes the effort to determine the shortest route to the destination but expect to reach their destinations in the shortest possible time. Thus, this system involves providing the shortest time taking path to their destination automatically. Optimization is achieved by splitting the entire process into two. First finding the shortest paths to the end station and then calculating and zero down to the final shortest time taking path. This approach is more generalized and is proved to offer higher efficiency and shorter process time then the conventional methodologies.

Keywords: Railway network; minimizing time; maximizing speed; Optimization**;** MAXSMINT;

1. Introduction

In this fast paced world, noting the possible routes and the time schedule of the trains available at the stations are not usually done by the passengers. This leads to them travelling the longer route or going to the wrong station. This instigates them to seek other means of transportation to reach the destination to avoid all these problems. Therefore, in the recent years some of the works by Prathap [1] *et al.,* and Brindha [2] *et al.,* has gone into the automation of the customer information especially the shortest path with respect to distance to the destination. The definition of our problem is as follows the passenger starting in a particular station X has to reach the destination station Y. For this problem, the arc length between two consecutive stations is a time depending function. That's to travel from station A→ B in time t at which the passenger starts at the station A determines the transfer time and based on this, the calculation of the shortest route proceeds. The overall goal here is to find a path such that by following this path, the passenger will reach the station at the shortest possible time.

The reason behind this technical implementation is the inspiration by different researches and their analysis. In some algorithm by Retvari [3] *et al.,* Zhu[4] *et al.,*shan[5] *et al., and Orda[6] et al.,* are determining the fastest travel time

* Brindha.G.R. Tel.: 9487755985;

E-mail address: brindha.gr@ict.sastra.edu.

1877-7058 © 2011 Published by Elsevier Ltd. Open access under CC BY-NC-ND license.

doi:10.1016/j.proeng.2012.01.914

between two nodes based on a given starting time. Cookie [7] *et al.,* investigated with dynamic programming technique to solve the problem which is time dependant. Subsequently, Soltani [8] *et al.,* Domenico[9] *et al.*, Misra[10] *et al.,* Solka[11] *et al.,* Sedgewick [12], *et al.,*Sherali [13] *et al.,* proved that applying Dijkstra's naming method to the time-expanded network is guaranteed to return an optimal solution. Trtiff [14], Rees [15] and Ali [16] *et al.,*have tackled different variation of the least cost shortest path. These includes, multiple shortest paths with single source in least cost and turn constraints. And also point to point connection problem, multicast trees with minimum cost are analyzed with TSP by Natu [17] *et al.,* Kosaraju [18] *et al.,*Shu Li [19] *et al.,*Kohn [20] *et al.,*Bellman [21] *et al.,*.

2. Requirement Analysis

For the functioning of the system, it is assumed that the entire map of that particular railway network is available. The time schedule at the individual stations is also assumed to be available. All the information regarding the functionalities of the trains is automatically updated by the information gathered from the main server network. The system now has all the information pertaining to the various stations in the network, the trains that are available (updated from the server), the junctions and the departure and arrival time of the trains at the various stations. The starting station and the time at which the passenger enters the start station are also provided by the passenger when they invoke this system to find the shortest route to their destination. All that is need by the passenger is to enter the destination and at that instant, the current time, the destination and the current station (initial station) is taken in by the system to calculate the shortest route. Figure 1-a shows the route map of a city and the stations are taken as nodes and the distance between the stations are taken as the arc and a directed graph is drawn and the time between the stations are provided since it is the important parameter in our algorithm. (Source: http://en.wikipedia.org/wiki/File:MRT_LRT_system_map_(current)_05-09.png-30/07/11)

Fig. 1. Route Map of a city

2.1 Time- expanded digraph

Time-expanded digraph method is adopted here where all the nodes are assumed to be consisting of clusters of arrival time and departure time. Since the passenger may change the train within a station, the arrival cluster should be connected to all the possible points in the departure cluster. For each station X(i), there are several arrival points A(x) and departure points D(y). The arrival and departure points correspond to the time of arrival and departure respectively.

2.2 Preliminaries and Node Assumptions

Each node is having a set of in-degree and out-degree nodes. In-degree nodes are also known as the arrival nodes which have their path arriving at the current node. The out-degree nodes are also known as the departure nodes which have their path departing from the current node. The in-degree and out-degree nodes correspond to the arrival and departure points for the nodes in the time-expanded digraph. Out degree based path determinations.

Since only the paths leading towards the goal node or destination station are to be considered, the goal node is taken as the start node and the path is figured out by using the depth first search algorithm as the base and by taking the out-degree nodes as the subsequent nodes for expansion. Here, as an example, only depth first search is done to determine the initial node from the goal node, but bidirectional search may also be used. Since the search space is finite, the efficiency would not be affected by using either of the two methods. Since all the possible paths from the initial to the final are to be determined the methods have to be modified to suit the purpose.

After the first step, which is done to determine the various paths which are leading from the starting to the ending node, the shortest time in all these paths are to be determined. This is done by using the time-expanded digraph, since the schedule of all the arrival and departure time is already present.

For a passenger, the shortest arrival time at the destination is required. Thus based on the requirement, the arrival time is taken as the basis of calculation of time between the nodes. The nearest arrival time at the next node $(n+1)^{th}$ is noted for which the departure time at the current node $(n)^{th}$ is greater than the arrival at the current node. Once the arrival time at the goal node is determined, pruning is done to eliminate the paths which may take a longer time then the current path, if another path is having a shorter time than the previous shortest arrival time to the destination, then the new path is taken and the arrival time at the goal node for that particular path is used for the pruning process in subsequent paths. This process is continued till the goal node is reached.

3. Algorithm

```
Begin
Preliminaries
For i=1 to total no. of stations
        Assign  Node id, Arrival_Time, Departure_Time  //Assign indegree, outdegree  for each node//
End for
Get Source_Node , Destination_Node and  Start_Time
Assign
        Destination_Node=Current_Node
        Nodes{}=insert _first(indegree (current_node))
Declare
        Fringe{},Non_Leading{};
        Level[m][n]={} where m-=1 to possible levels, n=1 to no. of nodes ;
Level Extration :
While(not empty)Nodes{}
{If (Current_Node is in Non_Leading{})
         remove from fringe
        break
Else
        Nodes{}=insert_first(indegree(current_node))
Endif
If(Nodes{i} \*first node*\= fringe{})
        remove node from Nodes{}
        break
endif
//Backtracking with DFS and pruning
 if (node{i}=Source_node)
        for level 1 to m
                for all nodes n in fringe{}
                        assign L[m][n]= fringe{}
        Break
endif
if(indegree(current_node)=0)
        remove node from fringe{}
```

```
        remove node from nodes{}
        remove node from indegree
        add node to n{}
        break
else
        current_node=first(nodes{})
        remove node from node{}
        assign node to first(fringe{})
endif
}
```

Time Based optimization with Prunning

```
Copy L'[m][n]=L[m][n]
Current_Node_Time=start_time
For each level m in L'[m][n]
For all node n
        If(n=Destination_node)
                Journey_time=Current_Node_time
                Break
        If Departure_time(n)>Arrival_time(n) //Compare (schedule,Current_Node_time)//
        Current_node=n+1
        If( Arrival_time≥Goal_time)
                Remove m
        Break
Display Journey_time
Display L'[m][n], the level m which leads to Journey_time (optimal)
```

In this algorithm, first, the inputs are accepted and the various nodes are fetched from the database. The inputs include the initial and final nodes and the time of departure. Then, the since the search is done from the destination to the initial node, the destination node is taken as the current node and the in-degree nodes (the nodes which leads to the current node) is taken in the nodes list inserted from the first. Next, the various routes to the destination are determined by the process of level extraction. Here the depth first search is used to search for the initial node from the destination through the in-degree nodes. The nodes which do not lead to the destination are placed in the non_leading list and are compared with the in-degree nodes to eliminate the nodes which do not lead to the destination. Thus by using this search algorithm the various routes to the destination maybe determined. Further the process is to determine the route which takes the shortest time. Thus, the first path is taken and the time taken is calculated. During the calculation of the time in the next path, if at any node the time taken till then exceeds the time taken by the first route, then the route is pruned. In the same manner the shortest time route to the destination is determined.

In Figure-1 the initial node and the final node is indicated in red colour. Based on that, Figure 2-a,b,c shows the possible paths from the source node to the destination node. Table-1 explains the possible path for above said nodes and the travelling time for that corresponding path. From that paths based on the optimum time taken by the path is selected, ie path 1 through which you can reach the destination with in 42 mins which is less than the other two paths.

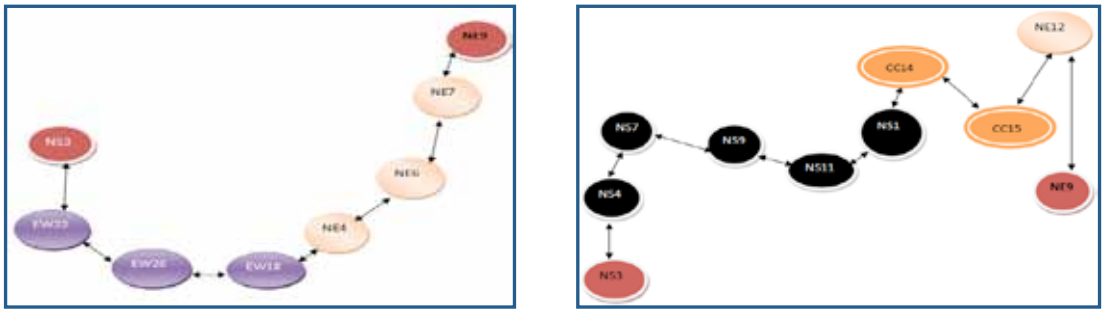

Fig. 2. (a) Possible path-1 from source to destination; (b) Possible path-2 from source to destination

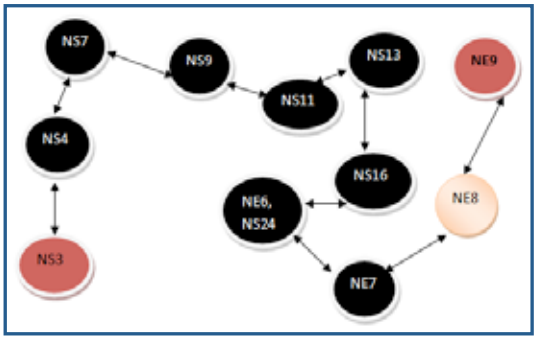

Fig. 2. (c)Possible path-3 from source to destination

Table-1 Possible Path Details with Time Duration

S.no	Nodes Travelled	Travel Time
1	***NS3,NS2,EW24-EW16,NE3-NE9***	***42*** Mins
2	NS3,NS4-NS17, CC15,CC14,CC13,NE12,NE10,NE9	44 Mins
3	NS3,NS4-NS24,NE6,NE7-NE9	52 Mins

4. System Architecture And Management

Fig.3. shows the functional diagram of MAXSMINT algorithm and Fig.4. depicts the state transition diagram for the technique MAXSMINT

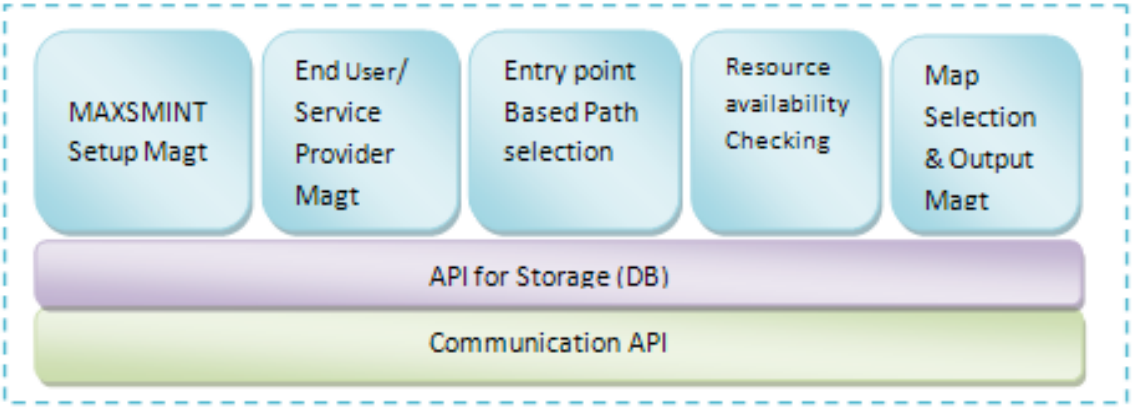

Fig. 3. Functional Diagram of MAXSMINT

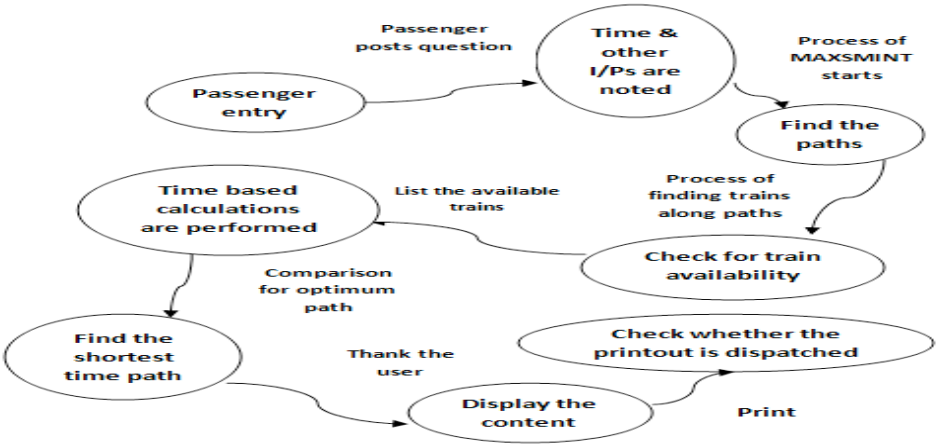

Fig. 4. State Transition Diagram

5. Simulation Analysis

In the user screen Fig.5. (a), the traveler should enter the date, time, starting station and the destination station. The algorithm will take the all possible combination for the nodes then from the acquired possibilities it will calculate the less travelling time taken by the route and that map will be displayed as shown in fig.5. (b). Fig. 6. shows the comparison chart of the time taken by the travelers who were used our MAXSMINT and those who were not used.

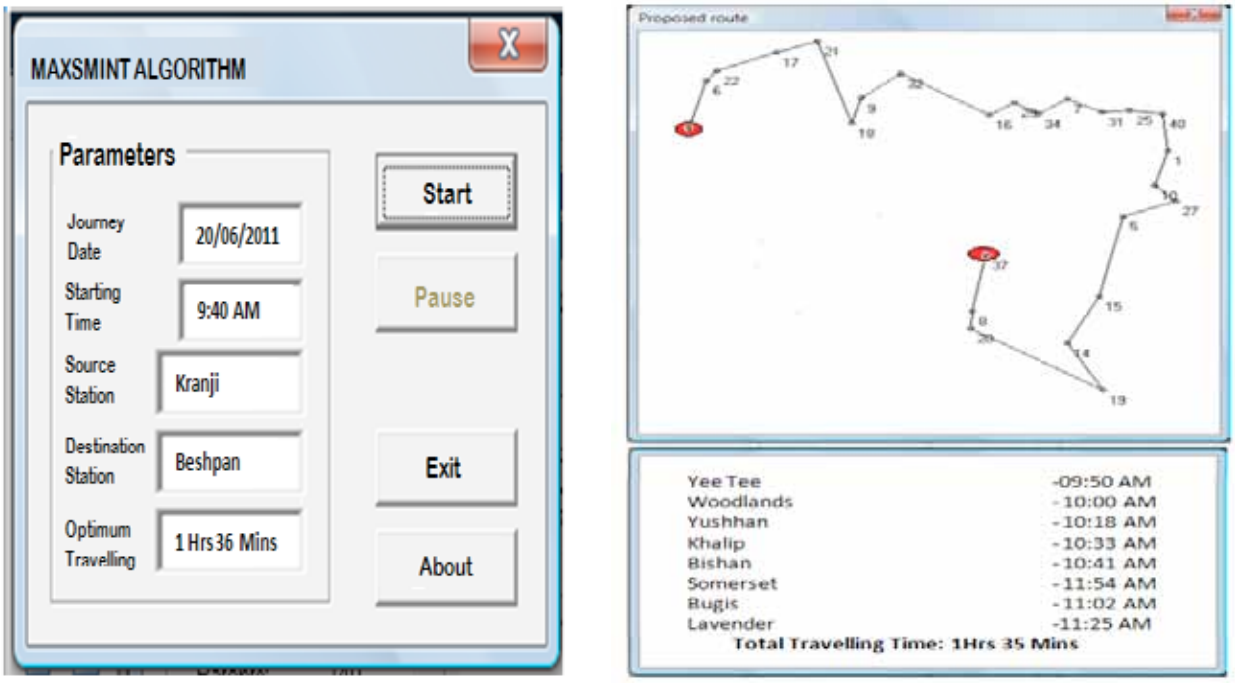

Fig. 5. (a) MAXSMINT Simulation GUI; (b) MAXSMINT Route Sheet

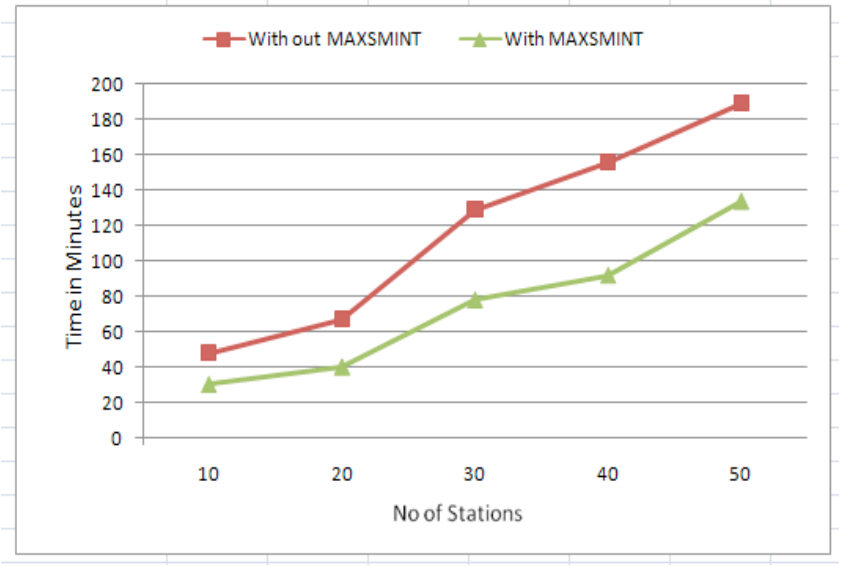

Fig. 6. Time consumption – With MAXSMINT & without MAXSMINT

6. Conclusion And Future Expansion

In this work, we have presented a novel technique which is applicable in a variety of real time scenarios. With the entire world under crunches for time, more and more emphasis has been laid on the optimizing and speeding up of process. With the implementation of techniques such as MAXSMINT, it can be observed how effectively the travelling time in railways is decreased and how the optimization in using railways is increased. We have also seen through practical realization of this algorithm that it is indeed suitable for realistic scenarios. In-order for the technique to evolve with the needs of the fast-paced world, the technique has inherent flexibility which allows it to be modified and gives scope for further enrichments. Dynamism maybe incorporated by communication being enabled through the phones of the travelers. When a particular traveler's phone number is linked with the system, any changes in the train schedule and the shortest route may be informed from time to time. Moreover, this technique of optimization is not only restricted to railways and can be extended to bus services also.

References

[1]. Joe Prathap, P.M., Brindha, G.R. :"An Innovative TESOR Algorithm for Perfect Routing", International conference on signal processing, communication, computing and network technologies (NCACNSP) 2011.
[2]. Joe Prathap, P.M., Brindha, G.R. :"A Revised TESOR Algorithm for Perfect Routing," First International conference Computer Science, Engineering and Information Technology (CCSEIT) 2011.[Accepted for Pubication]
[3]. Retvari, G., Biro, JJ., Cinkler, T. :"On Shortest Path Representation," Budapest Univ. of Technol. & Econ., Budapest, IEEE, vol.15, issue 6, Dec. 2007 , pp 1293, [Networking, IEEE/ACM Transactions]
[4]. Zhu, S., Huang, G.M. : "A new parallel and distributed shortest path algorithm for hierarchically clustered data networks", Nortel Inc., Richardson, TX, IEEE, vol 9, issue 9, September 1998 [Parallel and Distributed Systems, IEEE Transactions]
[5]. Garng Huang Shan Zhu. : "A new distributed shortest path algorithm for hierarchically clustered data networks", Dept. of Electr. Eng., Texas A&M Univ., College Station, TX, IEEE, vol 3, 21-23 Jun 1995 [American Control Conference, 1995. Proceedings]
[6]. Orda, A., Rom, R., "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length". Journal of the Association for Computing Machinery 37 (3), 607–625, 1990.
[7]. Cooke, K..L., Halsey, E. : "The shortest route through a network with time-dependent internodal transit times". Journal of Mathematical Analysis and Applications 14 (3), 493–498, 1966.
[8]. Soltani, A.R., Tawfik, H., Goulermas, J.Y., Fernando, T. : "Path planning in construction sites: performance evaluation of the Dijkstra, Ap, and GA search algorithms", Advanced Engineering Informatics ,291–303, Elsevier Ltd, 29 May 2003.
[9]. Domenico Cantone Simone Faro. :"Two-Levels-Greedy: a generalization of Dijkstra's shortest path algorithm", Electronic Notes in Discrete Mathematics 17 (2004) 81–86, Elsevier B.V, 2004.

[10]. Jayadev Misra. : "A walk over the shortest path: Dijkstra's Algorithm viewed as fixed-point computation", Information Processing Letters 77 (2001) 197–200, Elsevier Science, 2001.
[11]. Jeffrey Solka, L., James Perry, C., Brian Poellinger, R., George Rogers, W. : "Fast computation of optimal paths using a parallel Dijkstra algorithm with embedded constraints", Neurocomputing, 195-212, Elsevier Science , 10 February 1994.
[12]. Sedgewick, R., Vitter, J., "Shortest paths in Euclidean graphs". Algorithmica 1 (1), 31–48, 1986.
[13]. Sherali, H., Ozbay, K., Subramanian, S., "The time-dependent shortest pair of disjoint paths problem: complexity, models, and algorithms". Networks 31,259–272, 1998.
[14]. Jesper Larsson Trtiff. : "An experimental comparison of two distributed single-source shortest path algorithms, Parallel Computing", 1505-1532, Elsevier Science, April 1995.
[15]. Rees, W.G.: "Least-cost paths in mountainous terrain", Scott Polar Research Institute, University of Cambridge, Lensfield Road, Cambridge CB2 1ER, UK, Computers & Geosciences 30 (2004) 203–209, Elsevier Ltd, 2004 .
[16]. Ali Boroujerdi, Jeffrey Uhlmann, : "An efficient algorithm for computing least cost paths with turn constraints", Information Processing Letters 67 (1998) 317-321, Elsevier Science, November 1997
[17]. Madan Natu, Shu-Cherng Fang. :"The point-to-point connection problem - analysis and Algorithms", Operations Research und industrial Engineering, Discrete Applied Mathematics 78 (1997) 207- 226, Elsevier Science B.V, December 1996.
[18]. Kosaraju, S.R., Park, J.K., Stein, C. :"Long tours and short superstrings", IEEE Computer Society, 1994, pp. 166–177, [Proc. 35th Ann.Symp. on Foundations of Comput. Sci.,]
[19]. Shu Li, Rami Melhem, Taieb Znati, :"An efficient algorithm for constructing delay bounded minimum cost multicast trees", J. Parallel Distrib. Comput. 64 (2004) 1399 – 1413, Published by Elsevier Inc., August 2004
[20] Kohn, S., Gottlieb, A., Kohn, M. : "A Generating Function Approach to the Traveling Salesman Problem, ACM Press, 1997, pp. 294–300. [ACM Annual Conference]
[21]. Bellman, R. :"Combinatorial Processes and Dynamic Programming", in Bellman, R., Hall, M., Jr. (eds.), American Mathematical Society, ,2004, pp. 217–249 [Combinatorial Analysis, Proceedings of Symposia in Applied Mathematics]